Instituto Tecnológico de Aeronáutica

Relatório CES - 11
Lab 3: Ordenação

Matheus Felipe Ramos Borges

Turma 28.4

Prof. Armando Ramos Gouveia

Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica- ITA

Sumário

1 Introdução

O objetivo deste relatório é estudar o tempo de execução e a quantidade de comparações realizadas por diferentes algoritmos de ordenação: Bubble-Sort, Merge-Sort e Quick-Sort. Para a realização dos testes, foi utilizado o gerador de strings fornecido pelo Prof. Armando.

2 Ordenação em dois segundos

A tabela abaixo apresenta a quantidade de strings necessárias para que cada algoritmo atingisse aproximadamente 2 segundos de tempo de execução. v

Tabela 1: Número de entrada para 2 segundos de compilação em cada algoritmo.

Algoritmo	Entrada	Tempo (s)
Bubble-Sort	18.000	2,043
Merge-Sort	645.200	2,018
Merge-Sort Static	4.300.000	2,068
Quick-Sort	4.650.000	2,045

É possível observar a eficiência superior dos algoritmos Merge-Sort e Quick-Sort em comparação ao Bubble-Sort. Enquanto o Bubble-Sort já atinge 2 segundos com apenas 18 mil strings, o Quick-Sort precisa de 4,7 milhões de strings para atingir o mesmo tempo. Além disso, nota-se que o Merge-Sort na versão estática é mais rápido que sua versão padrão, e por isso será utilizado no restante do relatório.

3 Número de comparações e tempo de execução

As tabelas e gráficos a seguir referem-se ao número de comparações e ao tempo de execução para diferentes tamanhos de entrada dos algoritmos. Todos os gráficos são do tipo “dispersão com linhas”.

3.1 Bubble-Sort

Tabela 2: Número de comparações e tempo para o método Bubble-Sort.

Entrada	Comparações	Tempo (s)
1.000	499500	0,007
2.000	1999000	0,026
3.000	4498500	0,067
4.000	7998000	0,116
5.000	12497500	0,159
6.000	17997000	0,232
7.000	24496500	0,327
8.000	31996000	0,404
9.000	40495500	0,537
10.000	49995000	0,629

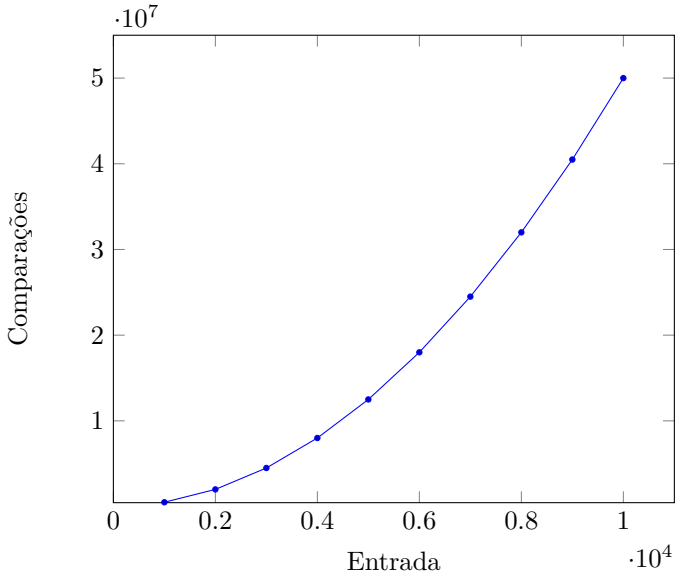

Gráfico 1: Número de comparações por número de entrada de strings no algoritmo Bubble-Sort.

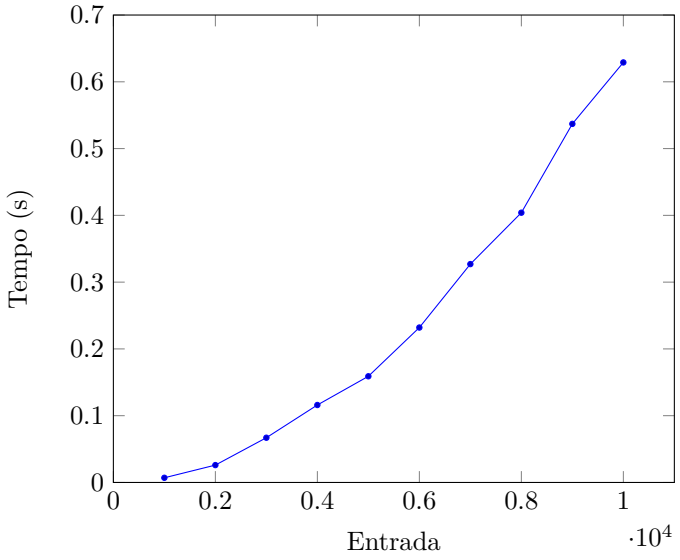

Gráfico 2: Tempo de execução por número de entradas no algoritmo Bubble-Sort.

3.2 Merge-Sort

Tabela 3: Número de comparações e tempo para o algoritmo Merge-Sort com o vetor static.

Entrada	Comparações	Tempo (s)
100.000	1536435	0,036
200.000	3272703	0,078
300.000	5084656	0,122
400.000	6945639	0,159
500.000	8836836	0,204
600.000	10769104	0,248
700.000	12723544	0,292
800.000	14691185	0,337
900.000	16678328	0,446
1.000.000	18673692	0,489

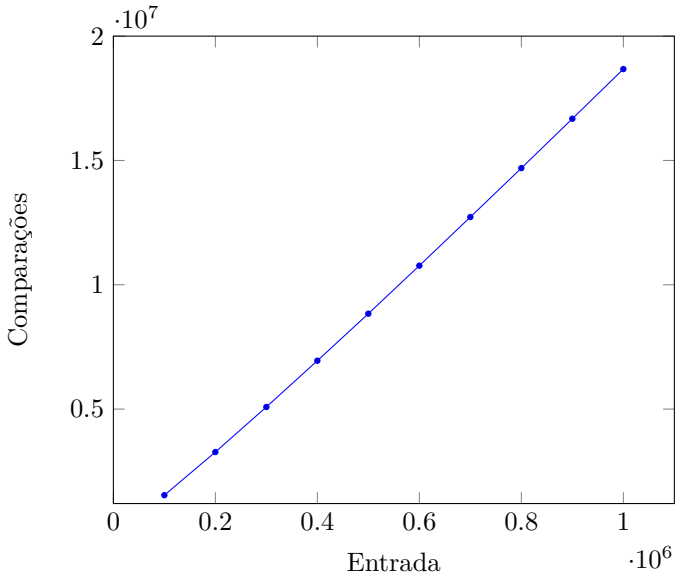

Gráfico 3: Número de comparações por número de entrada de strings no algoritmo Merge-Sort com o vetor static.

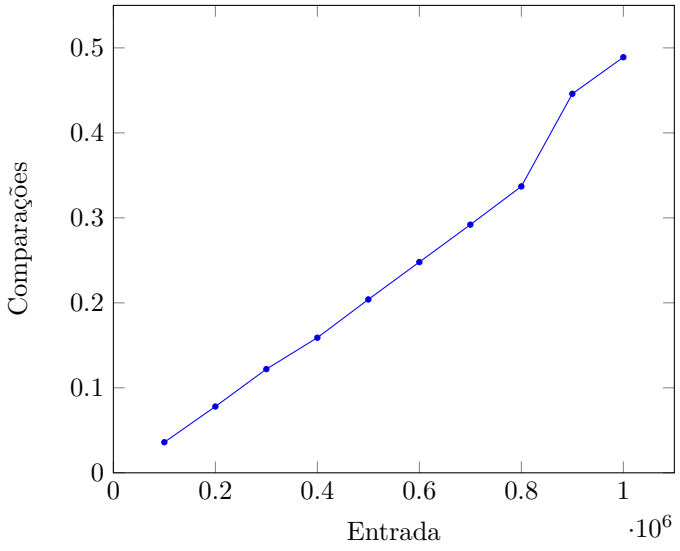

Gráfico 4: Tempo de execução por número de entradas no algoritmo Merge-Sort com o vetor static.

3.3 Quick-Sort

Tabela 4: Número de comparações e tempo para o algoritmo Quick-Sort.

Entrada	Comparações	Tempo (s)
100.000	2693268	0,028
200.000	5781807	0,059
300.000	9123080	0,100
400.000	12290403	0,120
500.000	15931049	0,164
600.000	19248738	0,209
700.000	22256950	0,213
800.000	26439862	0,293
900.000	29980700	0,344
1.000.000	34292874	0,336

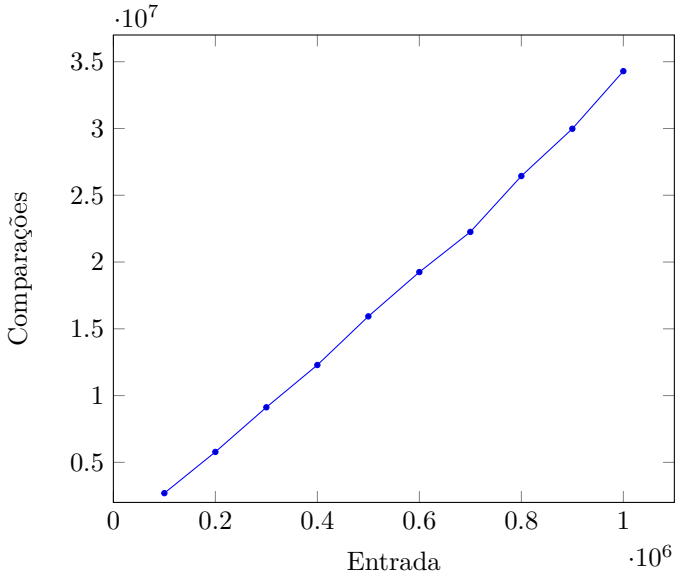

Gráfico 5: Número de comparações por número de entrada de strings no algoritmo Quick-Sort.

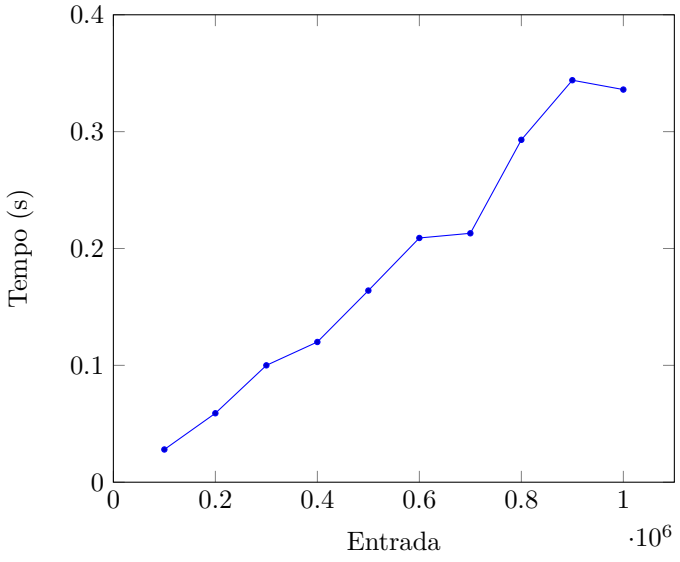

Gráfico 6: Tempo de execução por número de entradas no algoritmo Quick-Sort.

4 Análise dos Dados

A expectativa teórica para a complexidade dos algoritmos, em termos do número de comparações, é a seguinte:

- Bubble-Sort: $O(n^2)$
- Merge-Sort: $O(n \log n)$
- Quick-Sort: $O(n \log n)$

Espera-se que a análise de tempo também siga essas complexidades, uma vez que o tempo de execução deve ser proporcional ao número de comparações, sendo a parte que mais demora do código.

A partir dos resultados obtidos para o relatório, podemos observar que o algoritmo Bubble-Sort é significativamente menos eficiente que o Merge-Sort e o Quick-Sort. Enquanto o Bubble-Sort leva cerca de 2 segundos para ordenar 18 mil entradas, o Merge-Sort e o Quick-Sort só atingem esse tempo com mais de 4 milhões de entradas.

Ademais, embora o pior caso do Quick-Sort tenha uma complexidade pior que a do Merge-Sort, o comportamento do caso médio prevalece, tornando o Quick-Sort mais rápido, como evidenciam os dados obtidos. Apesar do tempo ser menor, o número de comparações no Quick-Sort é maior que no Merge-Sort, isso se deve a aleatoriedade na escolha do pivô — neste caso, sempre escolhido como o primeiro elemento —, o que faz com que nem sempre estava no caso mais otimizado, diferente do Merge-Sort, que sempre divide o vetor no meio, o que otimiza o processo.

Assim, os algoritmos apresentaram comportamentos condizentes com as expectativas teóricas. Nos gráficos do Apêndice 1, podemos comparar os pontos experimentais com as curvas de complexidade previstas para o número de comparações. Além disso, o Apêndice 2 mostra que o tempo de execução apresenta uma relação linear clara em função do número de comparações.

Dessa forma, os resultados obtidos estão condizentes com os valores teóricos, ajustando-se bem às curvas de n^2 e $n \log n$. A relação entre o tempo de execução e o número de comparações também é válida, demonstrando um comportamento linear à medida que o número de entradas vária.

5 Apêndice 1

Neste primeiro apêndice, mostro a comparação dos pontos obtidos no relatório com as respectivas curvas esperadas, obtidas pelo método da regressão linear. Para gerar os gráficos a seguir foi utilizado os logaritmos em base 10.

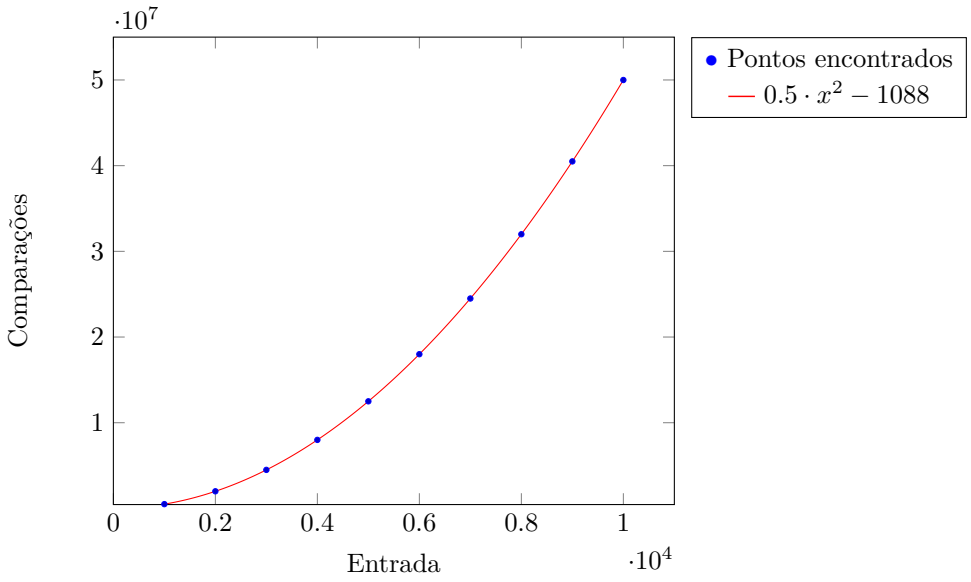

Gráfico 7: Número de comparações por número de entrada de strings no algoritmo Bubble-Sort em comparação com a curva esperada em vermelho.

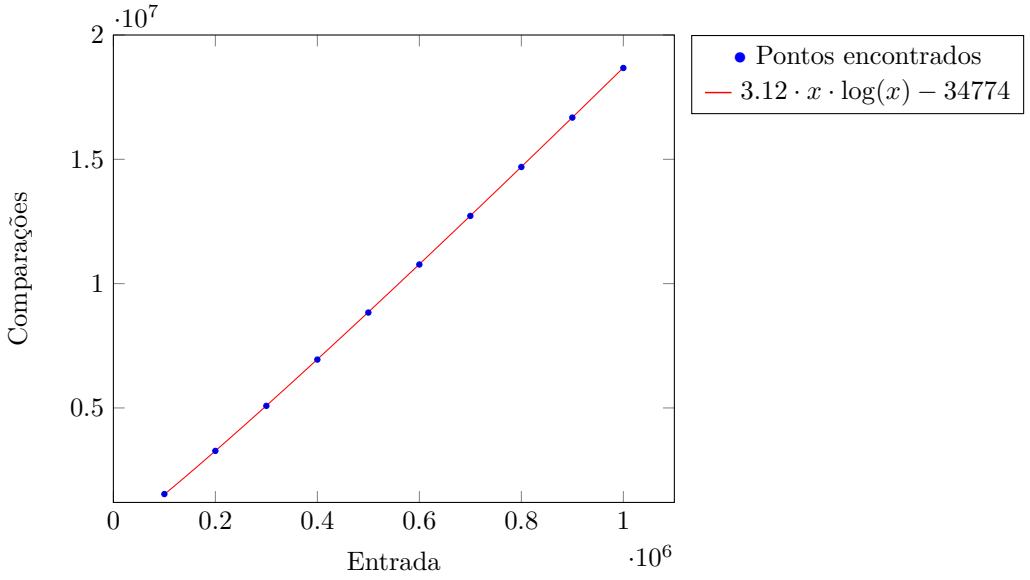

Gráfico 8: Número de comparações por número de entrada de strings no algoritmo Merge-Sort com o vetor static em comparação com a curva esperada em vermelho.

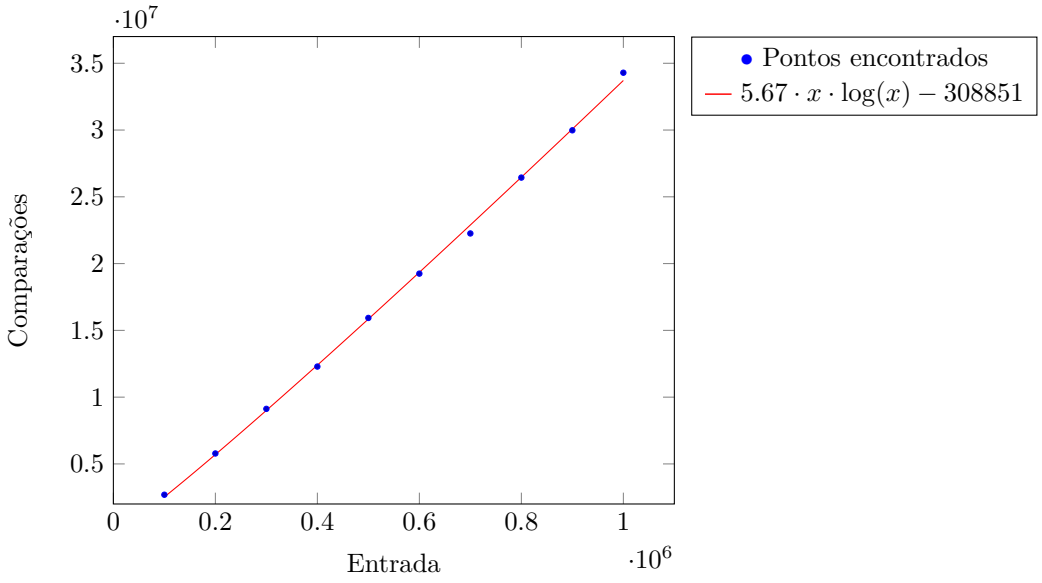

Gráfico 9: Número de comparações por número de entrada de strings no algoritmo Quick-Sort em comparação com a curva esperada em vermelho.

Pode-se ver, que os pontos encontrados se encaixam muito bem na complexidade teórica esperada.

6 Apêndice 2

Neste apêndice, foi feito uma breve análise da proporcionalidade entre o número de comparações e o tempo de compilação do programa.

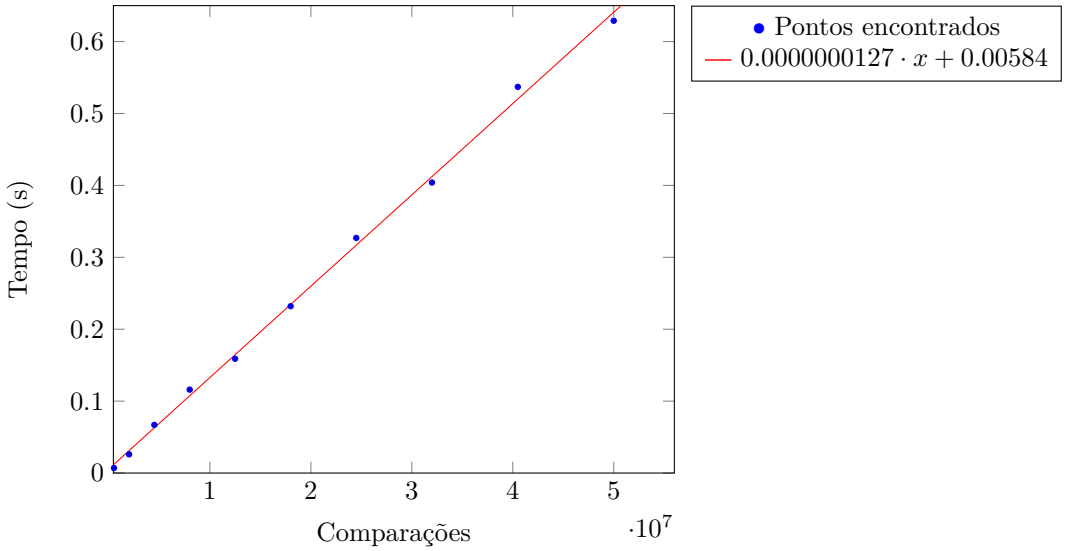

Gráfico 10: Relação linear entre comparações e tempo de execução do código no algoritmo Bubble-Sort.

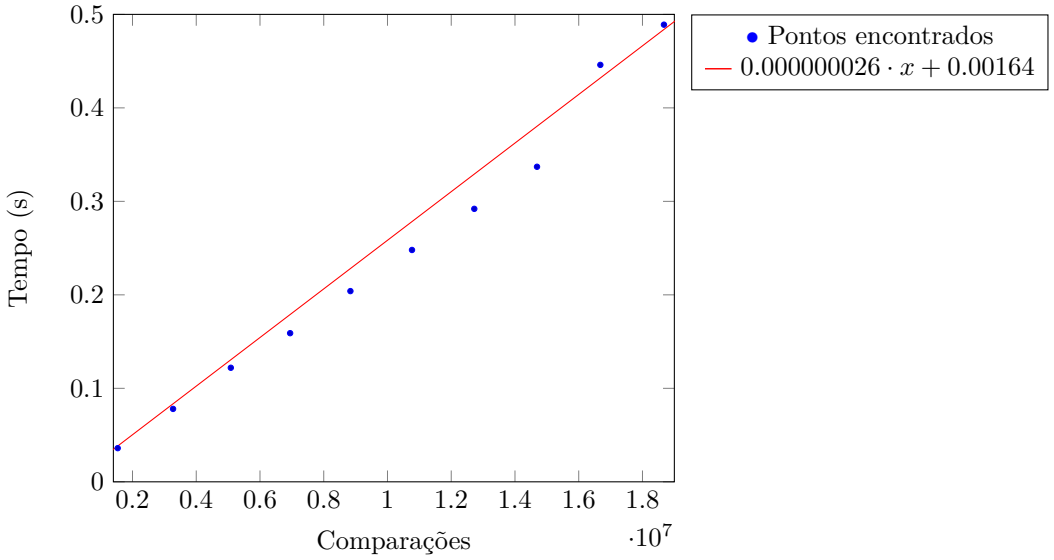

Gráfico 11: Relação linear entre comparações e tempo de execução do código no algoritmo Merge-Sort.

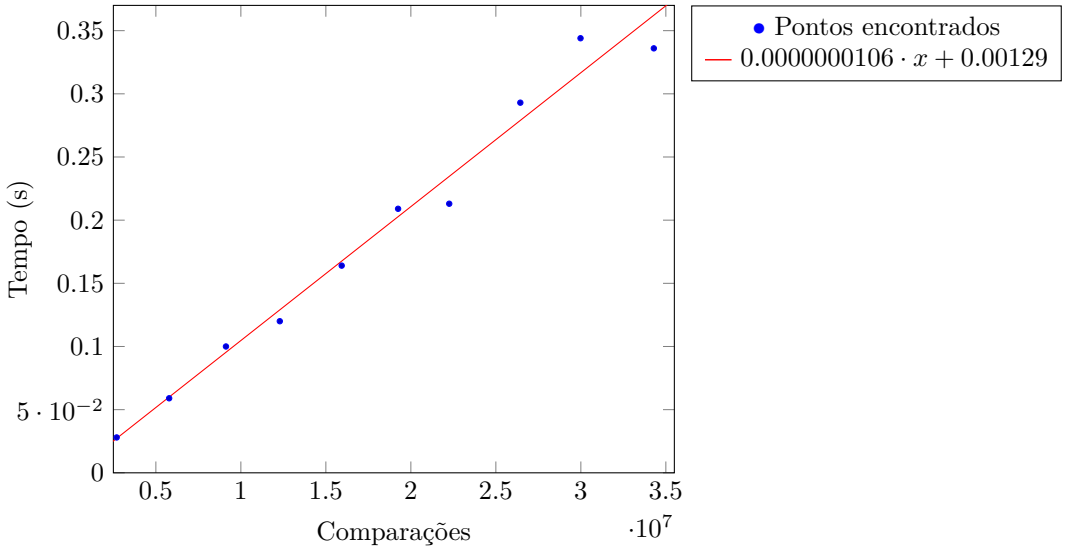

Gráfico 12: Relação linear entre comparações e tempo de execução do código no algoritmo Quick-Sort.

Desse modo, pode-se ver que esperar a mesma complexidade para o tempo e para as comparações é razoável, já que guardam uma relação de proporcionalidade.